THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 60, No. 8, October 1981
Printed in U.S.A.

Efficient Realization Techniques for Network Flow Patterns

By F. R. K. CHUNG, R. L. GRAHAM and F. K. HWANG

(Manuscript received January 20, 1981)

In this paper, we describe several new techniques for use in the design of switched communications networks. These techniques apply to the development of traffic routes which realize network traffic flows in the context of an existing optimization method that assigns these flows. The general ideas involve the careful selection of basic variables and the successive reduction of the problem to one of convex hull formation in Euclidean n-space and finding Hamiltonian circuits for a class of highly structured graphs. We include several examples showing how these techniques are applied.

I. INTRODUCTION

Recently, R. H. Cardwell[1] proposed a switched communications network design algorithm for the future stored program control network. The networks under consideration are nonhierarchical in structure and take advantage of traffic noncoincidence in routing. The basic objective of Cardwell's algorithm is to design a minimum cost trunking network which, by using an appropriate routing strategy, can carry the necessary traffic load and, at the same time, meet the required grade of service.

In this paper, we describe an extremely efficient method for producing an appropriate routing strategy. One of our original intentions was to develop a mathematical framework into which dynamic routing problems, such as those described later, could be placed. Indeed, it seems likely that the approach used here may be valuable for examining other classes of such routing problems.

II. BACKGROUND

In Fig. 1 we show a block diagram of Cardwell's algorithm. Suppose we wish to design a network using the algorithm. (See Ref. 1 for a

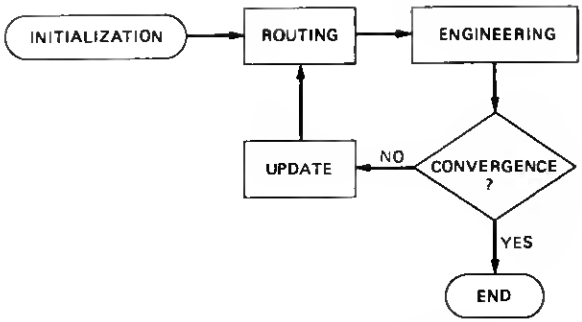

Fig. 1—Block diagram of Cardwell's algorithm.

more detailed description.) We start by initializing the blocking probabilities of each link. The routing module selects a set of the most economical paths for each pair of nodes and then assigns flow to the paths. Routes, which are ordered lists of paths, are then formed so that the probability that all paths in any list are busy is small enough to meet the required grade of service. Then, by means of a linear programming formulation, the routing module determines a network flow which minimizes the total cost, considering link costs and traffic noncoincidence. In the engineering module, the Erlang loss formula is then used to fix the number of trunks required for each link.[2] In the update module the ECCS method of Truitt is used to help minimize the network cost.[3] The blocking probabilities for all links are then updated. The whole process is now iterated until satisfactory convergence is achieved.

Figure 2 shows a block diagram of the routing module for the unified algorithm. A basic feature of our method is that actual routes are not formed until convergence has been obtained in an earlier part of the unified algorithm. Only after this occurs does the routing realization submodule generate the routes and provide the appropriate routing strategy. Refer to the work of Murray and Wong which gives efficient heuristic algorithms for solving the linear programming problems in this module.[4] The upper bound module is a new addition which helps

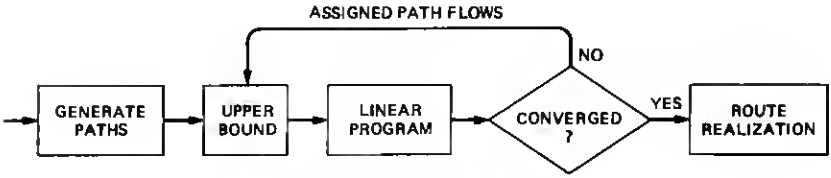

Fig. 2—Modified routing module.

the iterative procedure to converge more rapidly by setting stronger upper bounds for the carried loads of the various paths chosen.

One of the questions concerning this algorithm was the problem of synthesizing routes from the assigned path flows (the route realization block in Fig. 2). The following is a discussion of an efficient technique that accomplishes this.

III. CYCLIC ROUTING

Consider the special case of two nodes A and B. Assume there are n paths P_k, $1 \leq k \leq n$, between A and B. The amount of traffic to be carried on P_k is denoted by x_k, where we have normalized the traffic load so that one unit of traffic is attempted between A and B. The blocking probability for P_k is denoted by p_k. We will call the vector $\bar{x} = (x_1, \cdots, x_n)$ the desired traffic vector and $\bar{p} = (p_1, \cdots, p_n)$ the blocking probability vector. (The $\bar{x}$'s are actually outputs of the linear programming module.)

For a permutation π of $\{1, 2, \cdots, n\}$, by the route $R(\pi)$ generated by π, we mean the route in which the path $P_{\pi(1)}$ is tried and, if blocked, path $P_{\pi(2)}$ is tried. If that path is blocked, then path $P_{\pi(3)}$ is tried, etc.

The first question is: what are the traffic flows on the various P_k when route $R(\pi)$ is used? Let $q_k \equiv 1 - p_k$ and assume that π is the identity permutation, i.e., $\pi(k) = k$ for all k. Since one unit of traffic is initially attempted on P_1, the first path of $R(\pi)$, then q_1 units of traffic are carried on P_1 and p_1 units of traffic are blocked. These p_1 units are now attempted on P_2. Thus, p_1q_2 units get carried and p_1p_2 are blocked. Continuing this process, we see in general that on P_k, $p_1p_2 \cdots p_{k-1}q_k$ units of traffic are carried and $p_1p_2 \cdots p_{k-1}p_k$ are blocked. We condense this information into the flow vector $\mathbf{F}(\pi) = (F_1(\pi), F_2(\pi), \cdots, F_n(\pi)) = (q_1, p_1q_2, p_1p_2q_3, \cdots, p_1p_2 \cdots q_n)$. Note in particular that the amount of traffic which is blocked is just $p_1p_2 \cdots p_n$, independent of π.

The overall plan is to use each route $R(\pi)$ a certain fraction $\alpha(\pi)$ of the time, as π ranges over all permutations of $\{1, 2, \cdots, n\}$, so as to achieve the desired traffic flow x_k on each P_k. In other words, if possible, find $\alpha(\pi)$ with

$$\alpha(\pi) \geq 0, \sum_{\pi} \alpha(\pi) = 1$$

so that

$$\bar{x} = \sum_{\pi} \alpha(\pi)\, \mathbf{F}(\pi),$$

where π ranges over all permutations of $\{1, 2, \cdots, n\}$.

This is exactly the same problem as deciding whether $\bar{x}$ is in the

convex hull of points $\mathbf{F}(\pi)$ (considered as points in n-dimensional Euclidean space E^n) and, if so, finding a representation of $\bar{x}$ as a convex combination of the $\mathbf{F}_{(\pi)}$. Note that all the $\mathbf{F}(\pi)$'s are extreme points of the convex hull. Since

$$\sum_k \mathbf{F}_k(\pi) = 1 - p_1 p_2 \cdots p_n \text{ for any } \pi,$$

then tbe convex hull is actually (at most) an $(n-1)$-dimensional polytope. Thus, any point in the convex hull can be represented as a convex combination of some choice of n extreme points $\mathbf{F}(\pi)$.

As an example, we consider in detail the case $n = 3$. In Table I, we list the six possible π's and the corresponding $\mathbf{F}(\pi)$'s.

We will denote tbe permutation π which sends i to $\pi(i)$ by the sequence $\pi(1)\pi(2) \cdots \pi(\mathrm{n})$. This should not be confused with the ordinary cycle notation for a permutation π (which will also be used). For example, the permutation of {1, 2, 3, 4, 5, 6} given by $\pi(1) = 3$, $\pi(2) = 5$, $\pi(3) = 6$, $\pi(4) = 4$, $\pi(5) = 2$, $\pi(6) = 1$ can be written both as $\pi = (136)(25)(4)$ and $\pi = 3\ 5\ 6\ 4\ 2\ 1$.

Figure 3 shows a typical picture when these points are plotted in E^3. All six points lie on the plane $F_1 + F_2 + F_3 = 1 - p_1 p_2 p_3$. We should note here that we always assume $0 < p_k < 1$ for all k, since any path with blocking probability one can be removed without affecting the traffic flow, and any path which carries any traffic at all has positive blocking probability (less than one).

In general, we would like to be able to decide if the desired traffic vector $\bar{x}$ lies in the convex hull of $\mathbf{F}(\pi)$ and, if so, how to represent it as a convex combination of $\mathbf{F}(\pi)$. A natural choice to consider is a cyclic set of routes. For example, suppose we consider the three routes:

$$\pi_1 = 1\ 2\ 3,$$

$$\pi_2 = 2\ 3\ 1,$$

$$\pi_3 = 3\ 1\ 2.$$

Let us determine whether $\bar{x}$ is in the convex hull of these three points. Of course, a necessary condition is $\sum_i x_i = 1 - p_1 p_2 p_3$. In any case,

Table I—Flow vectors for the Case $n = 3$

π	$\mathbf{F}(\pi)$
1 2 3	$(q_1, p_1 q_2, p_1 p_2 q_3)$
1 3 2	$(q_1, p_1 q_2 p_3, p_1 q_3)$
2 1 3	$(q_1 p_2, q_2, p_1 p_2 q_3)$
2 3 1	$(q_1 p_2 p_3, q_2, p_2 q_3)$
3 1 2	$(q_1 p_3, p_1 q_2 p_3, q_3)$
3 2 1	$(q_1 p_2 p_3, q_2 p_3, q_3)$

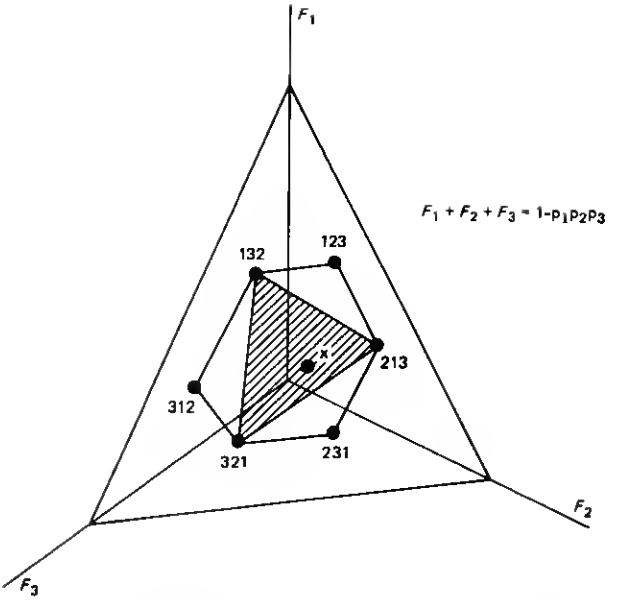

Fig. 3—Geometrical representation of the flow vectors for the Case $n = 3$.

since the convex hull is 2-dimensional, any point in it is a convex combination of some set of three extreme points. The cyclic sets seem reasonable choices since they apparently span rather large portions of the convex hull, although certainly not all of it. For example, in Fig. 3 we have shaded the convex hull of $F(\pi_1)$, $F(\pi_2)$, and $F(\pi_3)$. This is much larger than, say, the triangle spanned by $F(123)$, $F(132)$ and $F(231)$.

Therefore, we are looking for coefficients α_i such that

$$\sum_{i=1}^{3} \alpha_i \mathbf{F}(\pi_i) = \bar{x}, \tag{1}$$

with

$$\alpha_i \geq 0, \sum_i \alpha_i = 1.$$

By eq. (1), the α_i must satisfy

$$\sum_{i=1}^{3} \alpha_i F_k(\pi_i) = x_k, \; k = 1, 2, 3.$$

Expanding these equations using Table I, we obtain

$$\begin{aligned} \alpha_1 q_1 + \alpha_2 q_1 p_2 p_3 + \alpha_3 q_1 p_3 &= x_1, \\ \alpha_1 p_1 q_2 + \alpha_2 q_2 + \alpha_3 p_1 q_2 p_3 &= x_2, \\ \alpha_1 p_1 p_2 q_3 + \alpha_2 p_2 q_3 + \alpha_3 q_3 &= x_3. \end{aligned} \tag{2}$$

The determinant Δ of the system eq. (2) is given by

$$\Delta = \begin{vmatrix} q_1 & q_1p_2p_3 & q_1p_3 \\ p_1q_2 & q_2 & p_1q_2q_3 \\ p_1p_2q_3 & p_2q_3 & q_3 \end{vmatrix}$$

$$= q_1q_2q_3 \begin{vmatrix} 1 & p_2p_3 & p_3 \\ p_1 & 1 & p_1p_3 \\ p_1p_2 & p_2 & 1 \end{vmatrix}$$

$$= q_1q_2q_3(1 - p_1p_2p_3)^2.$$

Solving for the α_k, we have

$$\alpha_1 = \frac{1}{\Delta} \begin{vmatrix} x_1 & q_1p_2p_3 & q_1p_3 \\ x_2 & q_2 & p_1q_2q_3 \\ x_3 & p_2q_3 & q_3 \end{vmatrix}$$

$$= [(x_1/q_1) - (p_3x_3/q_3)]/(1 - p_1p_2p_3),$$

$$\alpha_2 = [(x_2/q_2) - (p_1x_1/q_1)]/(1 - p_1p_2p_3),$$

$$\alpha_3 = [(x_3/q_3) - (p_2x_2/q_2)]/(1 - p_1p_2p_3).$$

Letting

$$\sigma_k = \frac{x_k}{q_k}, \ \delta_k = \frac{p_kx_k}{q_k},$$

we see that the α_k are ≥ 0 if $\sigma_1 - \delta_3 \geq 0$, $\sigma_2 - \delta_1 \geq 0$, $\sigma_3 - \delta_2 \geq 0$.

For general n, a similar calculation shows that the corresponding system of n equations has determinant Δ given by

$$\Delta = q_1q_2 \cdots q_n(1 - p_1p_2 \cdots p_n)^{n-1}$$

and coefficient values

$$\alpha_{k+1} = [(x_{k+1}/q_{k+1}) - (p_kx_k/q_k)]/(1 - p_1p_2 \cdots p_n)$$

for the cyclic set of routes

$$\begin{array}{l} 1\,2\,3\,4 \cdots n \\ 2\,3\,4 \cdots n\,1 \\ 3\,4 \cdots n\,1\,2 \\ \quad\vdots \\ n\,1\,2 \cdots n-1 \end{array}$$

where addition of indices is modulo n, i.e.,

$$\alpha = (\sigma_1 - \delta_n)/(1 - p_1 \cdots p_n).$$

Consequently, we succeed with this cyclic choice of routes if all the α_k's are at least 0, i.e.,

$$\sigma_2 \geq \delta_1,\ \sigma_3 \geq \delta_2,\ \cdots,\ \sigma_n \geq \delta_{n-1},\ \sigma_1 \geq \delta_n. \tag{3}$$

Note that $\sum_i \alpha_i = 1$ follows at once from

$$\sum_i x_i = 1 - p_1 \cdots p_n$$

and, in particular, note that the labeling of the P_k is arbitrary. Any arrangement of the σ's and δ's satisfying eq. (2) will give us a cyclic set of routes which works, i.e., a set of routes which contains $\bar{x}$ in the convex hull.

In order to find these efficiently, we can do the following: From the given x_k and p_k form

$$q_k = 1 - p_k,$$

$$\sigma_k = \frac{x_k}{q_k},$$

$$\delta_k = \frac{p_k x_k}{q_k}.$$

We are just searching for a cyclic permutation $(j_1, j_2, \cdots, j_n)$ of $\{1, 2, \cdots, n\}$ such that

$$\sigma_{j_2} \geq \delta_{j_1},\ \sigma_{j_3} \geq \delta_{j_2},\ \cdots,\ \sigma_{j_1} \geq \delta_{j_n}.$$

To find this, form the directed graph G which has as its vertex set the set of paths P_k and an edge from each P_i to P_j for which $\sigma_j \geq \delta_i$. If in G we find a Hamiltonian circuit (i.e., a circuit passing through each vertex exactly once), say, $P_{j_1}P_{j_2} \cdots P_{j_n}(P_{j_1})$, then by the definition of the edges of G, we must have

$$\sigma_{j_2} \geq \delta_{j_1},\ \sigma_{j_3} \geq \delta_{j_2},\ \cdots,\ \sigma_{j_1} \geq \delta_{j_n},$$

which is precisely what we want.

Thus, we have shown that $\bar{x}$ can be realized from a cyclic set of routes if and only if G has a Hamiltonian circuit. Of course, the problem of finding a Hamiltonian circuit in an arbitrary graph is known to be an *NP*-complete problem (see Ref. 5 for an exposition of this term) and, therefore, almost certainly computationally intractable as the graph becomes large. Fortunately, however, the graphs G are far from arbitrary and, in fact, we can provide an algorithm for finding Hamiltonian circuits in them which runs in time $O(n \log n)$.

First, we may assume without loss of generality (by a suitable relabeling) that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n.$$

Note that a necessary condition for the existence of a Hamiltonian circuit in G is:

For all k, $2 \le k \le n$,

$$|\{i:\sigma_k \ge \delta_i\}| \ge n - k + 2, \tag{4}$$

where $|X|$ denotes the cardinality of the set X. To see this, note that if G has a Hamiltonian circuit, then for each k, there is at least one edge from a vertex in $\{P_k, P_{k+1}, \cdots, P_n\}$ to one in $\{P_1, P_2, \cdots, P_{k-1}\}$. Thus,

$$\sigma_k \ge \sigma_{t'} \ge \delta_{t''}$$

for some t', t'' with $t' \ge k > t''$. Therefore,

$$\{i:\sigma_k \ge \delta_i\} \supseteq \{i:i \ge k\} \cup \{t''\},$$

which implies

$$|\{i:\sigma_k \ge \delta_i\}| \ge n - k + 2.$$

In fact, eq. (4) is also a sufficient condition for G to be Hamiltonian. This can be seen from the following proof (by induction on n).

Suppose $n = 2$ and eq. (4) holds. Then clearly $\sigma_2 \ge \delta_1$ and G is Hamiltonian. Next, assume that eq. (4) is sufficient for all such graphs with $n - 1$ vertices. Suppose G has n vertices and satisfies eq. (4). Let G' be the induced subgraph on $\{P_1, P_2, \cdots, P_{n-1}\}$ (where we have assumed as usual that $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_n$). It is easy to see that G' also satisfies eq. (4). By induction, G' has a Hamiltonian circuit, say, $P_{j_1}P_{j_2} \cdots P_{j_{n-1}}$. Since G satisfies eq. (4),

$$\sigma_n \ge \delta_{j_i} \text{ for some } i,\ 1 \le i \le n - 1.$$

But also

$$\sigma_k \ge \sigma_n \ge \delta_n \text{ for all } k,\ 1 \le k \le n - 1.$$

Thus,

$$P_{j_1} \cdots P_{j_i} P_n P_{j_{i+1}} \cdots P_{j_{n-1}}$$

is a Hamiltonian circuit in G and the induction step is completed. This proves that eq. (4) is, in fact, a necessary and sufficient condition for G to have a Hamiltonian circuit.

We summarize the preceding discussion in the following.

Algorithm for cyclic routing

Inputs: Paths $P_1, \cdots, P_n$ joining two given points A and B, and the corresponding blocking probability vector $\bar{p} = (p_1, \cdots, p_n)$ and desired traffic vector $\bar{x} = (x_1, \cdots, x_n)$.

Object: To find a permutation π of $\{1, 2, \cdots, n\}$ satisfying

$$\bar{x} = \sum_{i=1}^{n} \alpha_i F(\pi_i)$$

with

$$\alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i = 1,$$

where F is the flow vector function and π_i is the cyclic route i, $\pi(i)$, $\pi^{(2)}(i)$, $\cdots$, $\pi^{(n-1)}(i)$ (i.e., P_i is tried first, then $P_{\pi(i)}$, etc.).

Algorithm:

(*i*) Calculate

$$\sigma_k = \frac{x_k}{1 - p_k} \text{ and } \delta_k = \frac{p_k x_k}{1 - p_k} \text{ for } 1 \leq k \leq n.$$

(Recall that we are always assuming that $0 < p_k < 1$ for all k.)

(*ii*) Relabel the σ_k, if necessary, so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$.

(*iii*) Set $\pi \leftarrow (1)$, $i \leftarrow 2$, $y \leftarrow \delta_1$, $z \leftarrow 1$.

(*iv*) If $\sigma_i < y$, go to (*vii*). If $\sigma_i \geq y$, insert i after z in the cycle representation of π.

(*v*) If $y > \delta_i$, set $y \leftarrow \delta_i$, $z \leftarrow i$. If $y \leq \delta_i$, y and z are unchanged. If $i < n$, set $i \leftarrow i + 1$ and go to (*iv*).

(*vi*) The desired Hamiltonian circuit is $P_1 P_{\pi(1)} P_{\pi^{(2)}(1)} \cdots P_{\pi^{(n-1)}(1)}$. Define

$$\alpha_k = \frac{\sigma_k - \delta_{\pi^{-1}(k)}}{1 - p_1 p_2 \cdots p_n}, \quad 1 \leq k \leq n.$$

$\bar{x}$ can be realized by using route π_k for the fraction α_k of the time, $1 \leq k \leq n$.

(*vii*) $\bar{x}$ cannot be realized by any set of cyclic routes.

End.

Note that except for (*ii*), in which $n \log_2 n$ operations are required in the ordering of the n σ_i's, all other steps require at most $O(n)$ operations. Thus, the computational complexity of the algorithm is $n \log_2 n + O(n)$ in time and $O(n)$ in space.

We point out that the desired traffic flow vector $\bar{x}$ can often be realized by more than one set of cyclic routes, i.e., the graph G might have more than one Hamiltonian circuit (each of which corresponds to a cyclic routing realization). The preceding algorithm will always produce one such realization provided any exists at all.

Two examples

Example 1: There are five paths between two points A and B. The desired traffic vector $\bar{x}$ and the blocking probability vector $\bar{p}$ are as follows:

$$\bar{x} = (0.185, 0.231, 0.220, 0.072, 0.242)$$

$$\bar{p} = (0.8, 0.7, 0.6, 0.5, 0.3).$$

Thus,

$$\bar{\sigma} = (0.924, 0.770, 0.550, 0.144, 0.346)$$

$$\bar{\delta} = (0.740, 0.539, 0.330, 0.072, 0.104).$$

The corresponding graph G is shown in Fig. 4.

From the algorithm, we find the Hamiltonian circuit, $P_1P_2P_3P_4P_5$, corresponding to the permutation $\pi = (12354)$. Therefore, we have the values shown in Table II.

The routing strategy is to use route π_i for the fraction α_i of the time.

Example 2: There are also five paths between A and B. However, the desired traffic vector $\bar{x}'$ and the blocking probability vector $\bar{p}'$ are slightly different from those in Example 1.

$$\bar{x}' = (0.191, 0.231, 0.220, 0.072, 0.242)$$

$$\bar{p}' = (0.7, 0.7, 0.6, 0.6, 0.3).$$

Thus,

$$\bar{\sigma}' = (0.638, 0.770, 0.550, 0.144, 0.346)$$

$$\bar{\delta}' = (0.445, 0.539, 0.330, 0.072, 0.104).$$

The corresponding graph G' is shown in Fig. 5.

From our algorithm, we find the Hamiltonian circuit, $P_2P_1P_3P_5P_4$, corresponding to the permutation $\pi' = (21354)$. Therefore, we have the values shown in Table III.

It is easily verified that

$$\bar{x}' = \sum_{i'=1}^{5} \alpha_{i'} \mathbf{F}(\pi_{i'}).$$

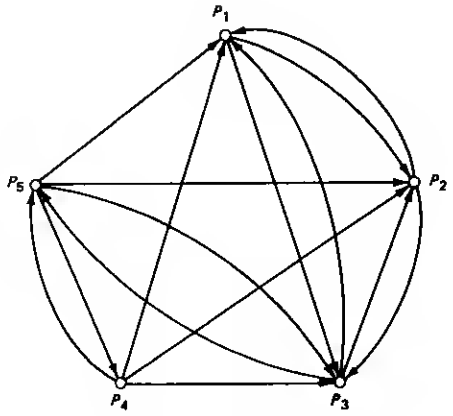

Fig. 4—Corresponding graph G for Example 1.

Table II—Values of coefficients for Example 1

i	Route π_i	Coefficient α_i
1	$P_1P_2P_3P_5P_4$	0.897
2	$P_2P_3P_5P_4P_1$	0.032
3	$P_3P_5P_4P_1P_2$	0.012
4	$P_4P_1P_2P_3P_5$	0.042
5	$P_5P_4P_1P_2P_3$	0.017

Note that there is another Hamiltonian circuit in G', namely, $P_1P_2P_3P_5P_4$, which gives an alternative cyclic routing realization as shown in Table IV.

Again, it is easy to verify that

$$\bar{x}' = \sum_{i=1}^{5} \alpha_i'' \mathbf{F}(\pi_i'').$$

IV. CYCLIC APPROXIMATIONS

As we mentioned earlier, the desired traffic vector $\bar{x}$ determined by the linear programming module can perhaps not be realized by a cyclic set of routes. In that case, we provide a routing strategy for approximating $\bar{x}$ by modifying our cyclic routing algorithm. This is most easily explained in terms of an example (this one was taken from data generated by a 28-point simulation of Cardwell[6]).

In this example, there are 8 paths from A to B. Table V shows the appropriate data.

Note that path 1 assumes the full traffic load, i.e., $x_1 = 1 - p_1$, which can be achieved if and only if every call requested is first attempted on path 1. In fact, this is a typical case of the existence of a least expensive direct line between two cities in a large toll switching network.

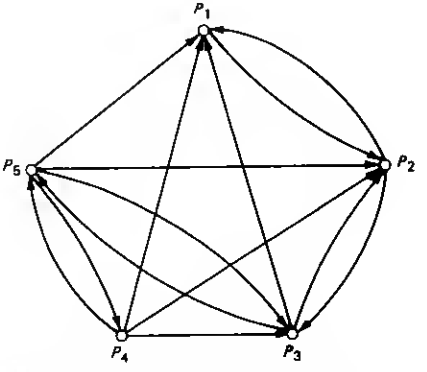

Fig. 5—Corresponding graph G' for Example 2.

Table III—Values of coefficients for Example 2

i	Route π_i'	Coefficient α_i'
1	$P_1P_3P_5P_4P_2$	0.103
2	$P_2P_1P_3P_5P_4$	0.730
3	$P_3P_5P_4P_2P_1$	0.109
4	$P_4P_2P_1P_3P_5$	0.042
5	$P_5P_4P_2P_1P_3$	0.017

There are several reasons why this $\bar{x}$ cannot be realized by cyclic routing. For example,

(i) Path 1 assumes the maximum possible traffic load, i.e., $x_1 = 1 - p_1$; this cannot happen with cyclic routing.

(ii) Traffic flow is highly unevenly distributed; in particular, paths 3, 6, and 8 get no traffic at all.

(iii) $\sum_i x_i \neq 1 - p_1 p_2 \cdots p_8$.

Let us form the graph G as described in the cyclic routing realization, namely, G has vertices $\{P_1, P_2, \cdots, P_8\}$ and there is a directed edge from P_i to P_j if $\delta_i \leq \sigma_j$ (see Fig. 6).

Here, G has no Hamiltonian circuit (and, in fact, is not even connected). In this case, we approximate $\bar{x}$ by taking a combination of (possibly trivial) disjoint circuits in G. The precise way this is done is described by the following algorithm.

Algorithm for approximate cyclic routing

(*i*) Calculate

$$\sigma_k = \frac{x_k}{1 - p_k} \text{ and } \delta_k = \frac{p_k x_k}{1 - p_k}, \; 1 \leq k \leq n.$$

(*ii*) Relabel the σ_k, if necessary, so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$. If $\sigma_n = 0$, define t by $\sigma_t > 0 = \sigma_{t+1}$, if $\sigma_n = 0$. Otherwise, define t to be n.

(*iii*) Set $\pi_1 \leftarrow 1$, $j \leftarrow 1$, $i \leftarrow 2$, $y \leftarrow \delta_1$, $z \leftarrow 1$.

(*iv*) If $y > \sigma_i$, go to (*vi*). If $y \leq \sigma_i$, insert i after z in the cycle representation of π_j.

Table IV—Alternative values of coefficients for Example 2

i	Route π_i''	Coefficient α_i''
1	$P_1P_2P_3P_5P_4$	0.591
2	$P_2P_3P_5P_4P_1$	0.339
3	$P_3P_5P_4P_1P_2$	0.012
4	$P_4P_1P_2P_3P_5$	0.042
5	$P_5P_4P_1P_2P_3$	0.017

Table V—Example not realized by cyclic routing

	$\bar{p}$	$\bar{x}$	$\bar{\sigma}$	$\bar{\delta}$	Approximation
1	0.30696	0.69304	1.00000	0.30696	0.69304
2	0.23850	0.00084	0.00111	0.00026	0.01832
3	0.30935	·	·	·	0.00233
4	0.45325	0.07555	0.13818	0.06263	0.06989
5	0.28685	0.13343	0.18710	0.05367	0.12343
6	0.33891	·	·	·	0.00116
7	0.60274	0.09685	0.24378	0.14694	0.08959
8	0.48781	·	·	·	0.00196

(*v*) If $y > \delta_i$, set $y \leftarrow \delta_i$, $z \leftarrow i$. If $y \leq \delta_i$, y and z are unchanged. If $i < t$, go to (*iv*). If $i = t$, go to (*vii*).

(*vi*) Set $j \leftarrow j + 1$, $\pi_j \leftarrow i$, $y \leftarrow \delta_i$, $z \leftarrow i$. If $i < t$, set $i \leftarrow i + 1$ and go to (*iv*). If $i = t$, go to (*vii*).

(*vii*) The routing strategy is given by using $\pi_1, \pi_2, \cdots, \pi_s$ as follows. Let

$$\beta_{i,k} = \sigma_k - \delta_{\pi_i^{-1}(k)},$$

$$\alpha_{i,k} = \frac{\beta_{i,k}}{\sum_j \beta_{i,j}} \text{ for } k \in \pi_i.$$

Use the route

$$\pi_{1,i_1}\pi_{2,i_2} \cdots \pi_{s,i_s}$$

for the fraction

$$\alpha_{1,i_1}\alpha_{2,i_2} \cdots \alpha_{s,i_s}$$

of the time where

$$\pi_{k,i_k} = [i_k, \pi_k(i_k), \pi_k^{(2)}(i_k), \cdots].$$

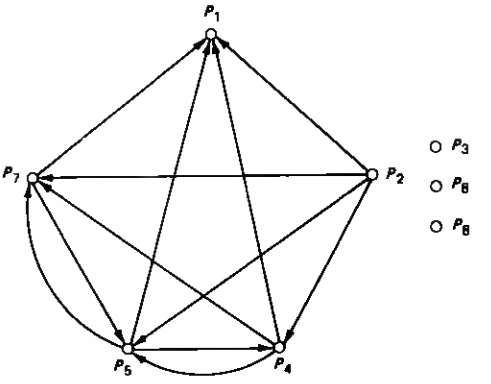

Fig. 6—Corresponding graph G for the Cardwell example.[6]

Continuing the example, we apply the above algorithm to the values in Table V. This results in the permutations $\pi_1 = (1)$, $\pi_2 = (547)$, and $\pi_3 = (2)$. The corresponding α's are given in Table VI. Note that we only use 5 of the 8 paths and the total blocking probability is 0.0057. If the blocking probability turns out to be too high to meet the required grade of service, we can make use of the remaining three paths in carrying the overflow by the modification shown in Table VII. The traffic flow generated by this routing strategy is listed in Table V under Approximation. Note that the approximation to the desired traffic flow is quite good.

As we pointed out before, one of the main reasons we cannot achieve the desired traffic flow exactly is that this is inherently impossible to do using convex combinations of the available routes, because of premature termination or inadequate constraints in the linear program. In the next section, we examine a method for correcting this difficulty.

Table VI—Values of coefficients for the Cardwell example

Route	Coefficient
$P_1P_5P_4P_7P_2$	$\alpha_{1,1}\alpha_{2,5}\alpha_{3,2} = 0.13132$
$P_1P_4P_7P_5P_2$	$\alpha_{1,1}\alpha_{2,4}\alpha_{3,2} = 0.27634$
$P_1P_7P_5P_4P_2$	$\alpha_{1,1}\alpha_{2,7}\alpha_{3,2} = 0.59234$

VII. UPPER BOUNDS

Again, we consider a set of paths $P_1, P_2, \cdots, P_n$ connecting two points and having blocking probabilities $p_1, p_2, \cdots, p_n$, respectively. The traffic flow on path P_i cannot exceed its capacity, namely, $1 - p_i$. Thus, an immediate upper bound on x_i for any realizable traffic flow vector $\bar{x}$ is

$$x_i \leq 1 - p_i \quad \text{for all} \quad i.$$

Similarly, for any two paths P_i and P_j, the total amount of traffic they can carry is $1 - p_ip_j$. Thus, if $\bar{x}$ is realizable then

$$x_i + x_j \leq 1 - p_ip_j.$$

Table VII—Values of coefficients for modified routes of the Cardwell example

Route	Coefficient
$P_1P_5P_4P_7P_2P_6P_8P_3$	0.13132
$P_1P_4P_7P_5P_2P_3P_6P_8$	0.27634
$P_1P_7P_5P_4P_2P_8P_3P_6$	0.59234

More generally, for any set of k indices $i_1, \cdots, i_k$, if $\bar{x}$ is realizable then

$$x_{i_1} + \cdots + x_{i_k} \leq 1 - p_{i_1} \cdots p_{i_k}. \tag{5a}$$

Furthermore, for $k = n$, eq. (5) must hold with equality, i.e.,

$$x_1 + \cdots + x_n = 1 - p_1 \cdots p_n. \tag{5b}$$

It is interesting to note that conditions (5a) and (5b) are also sufficient conditions for the realizability of $\bar{x}$ as a convex combination of flow vectors $\mathbf{F}(\pi)$. The proof is not difficult. Basically, it is as follows. Suppose $\bar{x}$ is an extreme point of the polytope ρ defined by the intersection of the half planes (5a) and the hyperplane (5b). Then $\bar{x}$ must satisfy at least one of the equalities in eq. 5a with equality, say without loss of generality.

$$x_1 + \cdots + x_r = 1 - p_1 \cdots p_r, \quad r < n. \tag{6}$$

We now use induction on n and express the point $\bar{x}' = (x_1 \cdots x_r)$ as a convex combination of the $r!$ flow vectors associated with paths $P_1, \cdots, P_r$. We next consider all the inequalities in eq. (5) which contain $x_1, \cdots, x_r$ as well as other x's. Typically, we might have

$$x_1 + \cdots + x_r + x_{j_1} + \cdots + x_{j_s} \leq 1 - p_1 \cdots p_r p_{j_1} \cdots p_{j_s}.$$

By eq. (6)

$$x_{j_1} + \cdots + x_{j_s} \leq p_1 \cdots p_r(1 - p_{j_1} \cdots p_{j_s}).$$

Again, we can use induction, this time on the new variables $y_k = x_k/p_1 \cdots p_r$, $r < k \leq n$, which satisfy the required analogues of eqs. (5a) and (5b). Finally, we piece together these two convex combinations to get the desired representation for $\bar{x}$. Since ρ is convex, then we are finished.

Of course, in actual practice some appropriate subset of the inequalities in eq. (5a) would be used in the upper bounding process (see Ref. 1).

VIII. CONCLUSIONS

In this paper we give necessary and sufficient conditions for determining whether a desired traffic flow vector (as specified by the linear programming solution portions of the algorithm) can be realized from a cyclic set of routes. The algorithm to verify the necessary and sufficient conditions can be implemented in $O(n \log n)$ time. When the conditions are not met, we propose an approximation method which uses several smaller cycles rather than a single cyclic set of routes.

In connection with the results we described earlier, it would be of interest to know what proportion of the volume of the polytope

spanned by the $n!$ flow vectors $\mathbf{F}(\pi)$ can in general be reached by *cyclic* routes. For $n = 3$, it seems that we can always cover at least ⅔ of the volume (actually, area in this case; see Figure 3). We have not yet examined the general case. In fact, we do not even know whether or not cyclic routes always span a positive fraction of the volume (independent of n).

IX. ACKNOWLEDGMENT

The authors have benefited greatly from discussions with G. Ash, R. Cardwell, V. Mummert, and other members of the Traffic Network Planning Department, who not only provided us with our first exposure to the general problem, but who were instrumental in extending and integrating the techniques we describe into an overall trunk network design algorithm which shows great promise.

REFERENCES

1. G. R. Ash, R. H. Cardwell, and R. P. Murray, "Design and Optimization of Networks With Dynamic Routing," B.S.T.J., this issue.
2. D. Bear, *Principles of Telecommunication Traffic Engineering*, London: Peter Peregrinus, Ltd., 1976, p. 38.
3. C. J. Truitt, "Traffic Engineering Techniques for Determining Trunk Requirements in Alternate Routing Trunk Networks," B.S.T.J., *33*, No. 2 (March 1954), pp. 277–302.
4. R. P. Murray and R. T. Wong, private communication.
5. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco: Freeman and Co., 1979.
6. R. H. Cardwell, unpublished work.